SE2 - A.Y. 2023-2024 - Analysis of a RASD Document

You are asked to read the RASD document available at
https://webeep.polimi.it/pluginfile.php/1116082/mod_folder/content/0/ProjectToBeReviewed/RASD.pdf

and fill in the questionnaire below highlighting the issues that you find relevant.
For your reference, the description of the assignment can be found here:
https://webeep.polimi.it/pluginfile.php/1116082/mod_folder/content/0/ProjectToBeReviewed/Assignment_2022-2023.pdf

The form can be answered by individuals or by groups of maximum 3 students. The information concerning the student submitting the form will be automatically collected by the system. Please make sure that the information you add in the form about your group mates (if any) are correct.

Students who registered for the R&DD project must do this work within their project group; it will be useful to understand how to develop an excellent RASD document. Other students can form groups as they prefer, also cross-class.

Preliminaries

Please make sure that the information you insert in the form is correct

1. Reference Professor *

- ( ) Prof. Di Nitto
- (•) Prof. Rossi
- ( ) Prof. Camilli

2. Is your group enrolled to the R&DD project? *

- (•) Yes
- ( ) No

3. Insert here the URL of the github repository you have created for your project. We will use that URL as an identifier of your group. *

4. How many group mates do you have? *

- ( ) No one, I am alone
- ( ) One
- (•) Two

Information about your first group mate

Please include here the information about your first group mate. The information about you are automatically recorded when you submit the form.

5. Last name of your first group mate *

6. First name of your first group mate *

7. Person ID of your first group mate *

Information about your other group mate

Please include here the information about your other group mate. The information about you are automatically recorded when you submit the form.

8. Last name of your other group mate *

9. First name of your other group mate *

10. Person ID of your other group mate *

Analysis of the RASD document

Now you are ready to answer to the questions concerning the RASD document

✓ **Esatto** 1/- Punti

11. Describe at most three between strengths and weaknesses you see in Sections 1 and 2. Be concise and clear. *

Weaknesses:
1. The goals are often ambiguous, the actors are not well specified, as often it refers to a general user without clearly defining who it is talking about (final user or operator). For goal G2, the meaning of the verb "charge" is taken for granted without referring to what is being charged. In addition, a few goals such as g1, g3, g4 and g6 are user actions, not goals that must be accomplished. Lastly, g6 is ambiguous as it does not specify when the proactive suggestion is raised.
2. Shared phenomena controlled by the machine SP10 to SP14 are not actually controlled by the machine: it's the user that initiates the action and the machine which responds to it. As an example, the user has to open the app in order to be shown the stations.
3. In the class diagram, the definition of relationships about the booking and recharge process is confusing. First, a booking can have multiple timeslots, but only one of these can be used to recharge. This is a questionable choice, as it needlessly complicates the management of what sockets are free or in use, and it is also not requested in the specification. There is also a discrepancy between the diagram and the text accompanying it, because the latter mentions that the booking should be connected to a set of sockets, while this is not present in the diagram at all (making the diagram wrong, as you cannot infer the number of used and free sockets per their type at all). In later pages, it is clarified that a timeslot can only be of 30 minutes (which begs the question of why there's even an end time at all), so multiple timeslots are needed in order to have a charging time of more than 30 minutes. This is still a confusing choice, as nowhere it is enforced that booked time slots refer to the same socket (if that relationship was supposed to be there), nor that the time slots are subsequent. Therefore, you can still book completely unrelated timeslots and only charge during some of these, as you can still only recharge once. The diagram also has many redundant relationships which form cycles, such as User -> Booking -> Recharge -> Payment -> User which allows to pay the same recharge multiple times.

✓ **Esatto** 1/- Punti

12. Describe at most three between strengths and weaknesses you see you see in Section 3. Be concise and clear. *

Strengths:
1. User interface graphics are clear, clean and appropriately represent the main functionalities of the program, although it could have been useful to show the booking screen.
Weaknesses:
2. The use-cases diagram is missing a lot of functionality related to searching and showing content. As an example, there is no function for the user to even see the list of nearby stations, to search for a specific station or to see the available sockets on a station.
3. All the responses in use-cases sequence diagrams look like invocations back to the user. Some user generated events are also represented with inappropriate components: for example in UC56 and UC7 the click of a confirmation link is modeled with an "alt confirmation email = true" and in UC9 the user clicking on a confirmation button is represented as "alt confirmationButton == true"

✓ **Esatto** 1/- Punti

13. Describe at most three between strengths and weaknesses you see in Section 4. Be concise and clear. *

Weaknesses:
1. The management of time slots is wrong, there is nothing preventing from having two overlapping timeslots booking the same socket. This can be seen by using the following fact:

```
assert canHaveOverlappingBooking {
   no disj b1, b2: Booking |
         some disj t1, t2: TimeSlot |
               b1.reservedSocket = b2.reservedSocket and
                     t1 in b1.timeSlots and
                     t2 in b2.timeSlots and
                     (not t1.startTime = t2.startTime) and
                     (gt [t1.startTime, t2.startTime]) and
                     (lt [t1.startTime, t2.endTime])
}
```

2. A booking can have multiple timeslots each with a different socket, which can also be different than the socket reserved by the booking itself:

```
assert bookingHasTimeslotsOfTheWrongSocket {
   no b: Booking, t: TimeSlot | t in b.timeSlots and
                        b.reservedSocket != t.socket
}
```

3. fact savedRechargeImpliesFullEffectiveInformation does not check that you have full effective information, only that you have the end time:

```
assert savedRechargeHasPartialInfo {
      no r: Recharge | r.rechargeStatus = Saved and
            r.startTime = none and r.endTime != none
}
```